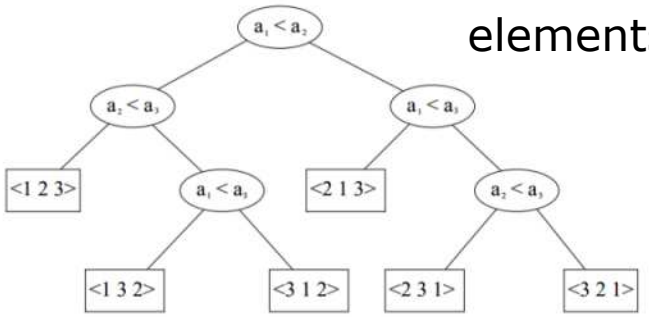
Illustrative Basic Examples

demonstrating the importance of

- models of computation
- problem specification
- limits of computability
- algorithmic optimality

Comparison -sort for 3 elements



Example 1: *Optimal Sorting Algorithm*

Problem specification:

Model of computation:

First algorithm:

Its cost analysis:

Second algorithm:

Its cost analysis:

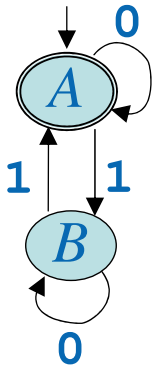
Proof of optimality:

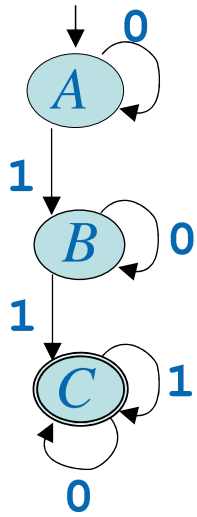
Example 2: Finite Automata

KAIST
CS422 M. Ziegler

demonstrating the *relevance* of

- models of computation and
- problem specification for
- limits of computability and
- algorithmic optimality





Theorem: a) $\{\ 0^n\ 1^n : n \in \mathbb{N}\ \}$ cannot be accepted by a finite automaton.

b) To every *non*-deterministic finite automaton there exists an equivalent deterministic one.

Example 3: Algebraic Complexity

KAIST CS422 M. Ziegler

n	$\log_2 n$ ·10s	$n\cdot\log n$ sec	n^2 msec	n^3 µsec	2^n nsec
10	33sec	33sec	0.1sec	1msec	1msec
100	≈1min	11min	10sec	1sec	40 Mrd. Y
1000	≈1.5min	≈3h	17min	17min	
10 000	≈2min	1.5 days	≈1 day	11 days	
100 000	≈2.5min	19 days	4 months	32 years	

Warmup Problem: Fix $n\in\mathbb{N}$. Given x, calculate x^n.

- Naïve algorithm: n-1 multiplications
- Improve: Calculate $x^2, x^4, x^8, \ldots, x^{2^k}$ for $k:=\lfloor\log_2 n\rfloor$
  Then multiply powers x^{2^j} with $b_j=1$, where
  $n=b_0+2b_1+4b_2+\ldots+2^k b_k$ is the binary expansion.
- Asymptot. optimality: Each multiplication at most doubles the degree of the intermediate results;
  so computing x^n requires at least $\log_2 n$ of them.

Example 4: Matrix Multiplication

KAIST CS422 M. Ziegler

- Input: entries of $n\times n$-matrices A,B — $O(n^3)$,
- Wanted: entries of $n\times n$-matrix $C := A+B$ — optimal
- High school: n^2 inner products á $O(n)$:

7 multiplications | +18 additions | of $(n/2)\times(n/2)$-matrizes

$$T_1:=(A_{2,1}+A_{2,2})\cdot B_{1,1}$$
$$T_2:=(A_{1,1}+A_{1,2})\cdot B_{2,2}$$
$$T_3:=A_{1,1}\cdot(B_{1,2}-B_{2,2})$$
$$T_4:=A_{2,2}\cdot(B_{2,1}-B_{1,1})$$

$$\begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}$$

$$C_{1,1}=T_5+T_4-T_2+T_7\ ,\quad C_{1,2}=T_3+T_2$$

$L(n) = 7\cdot L(\lceil n/2\rceil)$ — asymptotics dominated by #multiplications

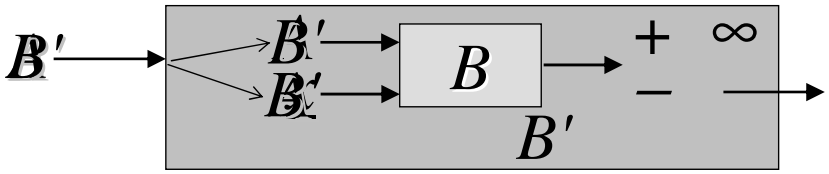
World record: $O(n^{2.37})$ [Coppersmith&Winograd'90, François Le Gall'14]

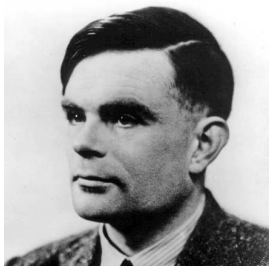
$L(n) = O(n^{\log_2 7})$, $\log_2 7 \approx 2{,}81$

Alan M. Turing 1936

KAIST
CS422 M. Ziegler

- first scientific calculations on digital computers
- *What are its fundamental limitations?*





- Undecidable Halting Problem *H*: **No** algorithm *B* can always correctly ans ~~simulator/interpreter *B* ?~~
  Given ⟨A,x⟩, does algorithm A terminate on input x?

Proof by contradiction: Consider algorithm *B'* that, on input *A*, executes *B* on ⟨*A*,*A*⟩ and, upon a positive answer, loops infinitely. How does *B'* behave on *B'* ?

Summary of §1

KAIST
CS422 M. Ziegler

The *Theory of Computation*

- considers mathematical models of computers,
- explores their capabilities and limitations
- as well as optimal asymptotic algorithmic cost
- for fully-specified computational problems.

We have seen basic examples:

- comparison-based branching trees
- finite automata
- algebraic unit-cost model (power, matrix mult)
- some (unspecific/generic) programming system